iSay: Invoking the Linux or OSX Speech Generation Commands from MOOS

June 2018

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139
project-pavlab/appdocs/app_isay

1 Overview

The iSay application is a way to invoke the native speech generation utilities of OSX (the say command) and GNU/Linux (the espeak command). It may also invoke the afplay commands available in both OSX and GNU/Linux to play a given wav or mp3 file. Normally, without MOOS running, one can generate voice from the command line with these built in commands:

```
$ say hello
```

in OSX, or in GNU/Linux with:

```
$ espeak hello
```

When iSay is running, the same can be accomplished by poking the MOOSDB:

```
$ uPokeDB SAY_MOOS=hello
```

The advantage being of course that the speech may be generated by an event in MOOS, e.g. a

vehicle returning. The iSay application also makes provisions for choosing the voice, the time in between utterances or in between utterance beginnings, and an utterance priority if a queue of utterances begins to form due to a backlog of several utterance requests. The utterance may be either a specified text of speech, or it may be a sound provide by a named `.wav` file. Presently, if a named `.wav` file is not found or if a named voice is not recognized, no sound will be generated. Future versions of iSay may make provisions for fallback sounds in these cases.

2 Configuration Parameters for iSay

The following parameters are defined for iSay. A more detailed description is provided in other parts of this section. Parameters having default values are indicated.

Listing 2.1: Configuration Parameters for iSay.

audio_dir:	Names a system directory to be added to path of directories to search for audio files named in a particular utterance request.
default_rate:	The speech rate to be used, in words per minute, if left unspecified for a particular utterance request (200).
default_voice:	The default voice to be used if left unspecified for a particular utterance request.
interval_policy:	Either `"from_start"` or `"from_end"`, indicating whether the interval of time between utterances is marked from the start of the previous utterance or the end of the previous utterance (`"from_end"`).
os_mode:	Either `"osx` or `"linux"` or `"both"`. The default is `"both"`
min_utter_interval:	The minimum time (in seconds) between utterances before a next queued utterance will commence (1.0).

3 Publications and Subscriptions of iSay

The interface for iSay, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ iSay --interface or -i
```

3.1 Variables Published by iSay

The only output of iSay is:

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility. Section 7.

3.2 Variables Subscribed for by iSay

The iSay application will subscribe for the following four MOOS variables:

- APPCAST_REQ: A request to generate and post a new apppcast report, with reporting criteria, and expiration. Section **??**.
- SAY_MOOS: A description of an utterance to be made. Section 4.
- SAY_FILTER: Either "none", "hold", or "ignore" ("none"). Section 6.
- SAY_VOLUME: Either "mute", "vsoft", "soft", "normal", "loud", "vlou", or a number in the range "[0,2]". This will only affect the volume of wav files only on MacOS, relative to the prevailing output volume on the computer. Text to speech (MacOS,Linux) and wav files on Linux, are not affected.

4 Specifying an Utterance

The iSay app works by receiving utterance requests through incoming MOOS mail via the SAY_MOOS variable. This mail has the following format by a couple examples:

```
SAY_MOOS = "say={hello world}, priority=100, voice=Albert, rate=250, source=henry"
SAY_MOOS = "file=file.wav"
```

The source component is optional and by default will be set to a string comprised of the MOOS community and the MOOS application that generated the utterance request, separated by a colon. See, for example, line 24 in Listing 2. This component provides an opportunity to be more specific about the source, as in line 25 in the same example.

The rate component specifies the speech rate in words per second. The default rate is 200. The priority component specifies the priority of the utterence request when the request is pushed onto the priority queue. See Section 5 for more on this.

The voice component allows one to request a voice different from the default voice. The options for this component depend on the options supported by the OSX say command and the GNU/Linux espeak command. These options may be found by typing "say -v ?" on the command line or "espeak --voices" on the command line in GNU/Linux.

5 The iSay Priority Queue

The iSay priority queue holds all received utterance requests before they are eventually popped and processed. The priority queue is popped (if non-empty) once on each iteration of iSay, and no more frequently than the allowed by the setting of min_utter_interval and the time it took to process the system command of the previous utterance.

By default, all uttterances have a priority of zero, and are simply processed first-come first-served. An utterance request may instead be specified with a higher (or lower) priority to affect the order in which it is popped from the queue. Requests of equal priority are processed first-come first-served. A request may also be made with "top" priority. These requests go immediatedly to the top of the priority queue. If multiple top priority requests are received, the latest one received will be handled first.

6 The iSay Filter

The iSay filter may be set to one of three values. The default is "none" and all utterances are received and processed normally. When the filter is set to "ignore", it's as if the mute button is hit. Any received utterances will be immediately put on the queue and any utterances popped from the queue will be handled according to the normal interval specifications. But the actual system command to generate the utterance will not be made. When the filter is set to "hold", all received utterances will be put in the queue and nothing will be popped from the queue until the hold is released. At some point the maximume queue size (100) may be exceeded.

7 Terminal and AppCast Output

The iSay application produces some useful information to the terminal on every iteration of the application. An example is shown in Listing 2 below. This application is also appcast enabled, meaning its reports are published to the MOOSDB and viewable from any uMAC application or pMarineViewer. See Section **??** for more on appcasting and viewing appcasts. The counter on the end of line 2 is incremented on each iteration of iSay, and serves a bit as a heartbeat indicator. The "0/0" also on line 2 indicates there are no configuration or run warnings detected.

The output in the below example comes from the example described in Section 8.

Listing 7.2: Example terminal or appcast output for iSay.

```
1  ============================================================================
2  iSay alpha                                                          0/0(96)
3  ============================================================================
4  Configuration Parameters:
5  -------------------------
6     Default Voice: alex
7      Default Rate: 200
8   Max Utter Queue: 100
9   Min Utter Inter: 1
10  Interval Policy: from_end
11
12 Status:
13 -------------------------
14   Utter Queue Size: 36
15   Unhandled Audios: 0
16              Filter: none
17
18 Source              Time  Time  Utterance
19                     Recd  Post
20 ------------------  ----  ----  ---------
21 alpha:uTimerScript  12.8  13.7  five
22 alpha:uTimerScript  1.4   12.1  four
23 alpha:uTimerScript  10.0  10.4  five
24 alpha:uTimerScript  7.7   8.7   five
25 james               1.2   7.1   three
26 alpha:uTimerScript  4.4   5.4   five
27 alpha:uTimerScript  1.2   3.9   two
28 alpha:uTimerScript  1.6   2.2   five
29 alpha:uTimerScript  0.7   0.7   one
```

```
30
31  ============================================================
32  Most Recent Events (8):
33  ============================================================
34  [14.33]: Utter Rec'd:say={nine}
35  [13.66]: Say:five
36  [13.46]: Utter Rec'd:say={eight}
37  [13.26]: Utter Rec'd:say={seven}
38  [12.96]: Utter Rec'd:say={six}
39  [12.76]: Utter Rec'd:say={five},priority=100
40  [12.66]: Utter Rec'd:say={four}
41  [12.66]: Utter Rec'd:say={three},source=james
```

The first few lines (4-10) show the configuration settings for iSay. The status of iSay is shown in Lines 12-16. In this case the queue is growing fairly quickly (line 14) since the utterance requests are being received about four times per second, but the minimum time between utterances is set to be one second (line 9). Processed utterances are shown in reverse order starting on line 21. This block has a maximum length of ten items, to keep the size of the AppCast report in check.

Recent events are shown in the block beginning on line 34. Events may be either a received or processed utterance. This block has a maximum length of eight items, to keep the size of the AppCast report in check.

8 A Simple Example

The below is a simple example, using the uTimerScript application to generate inputs to iSay. You can try this example by adding these to blocks to an existing example mission and adding iSay and uTimerScript to the Antler block. The script should generate spoken utterances of the numbers one through nine pretty much in sequence except that fives are handled with a high priority and spoken almost immediately. Since the min_utterance_interval is set to one second, fives are spoke almost half the time, and the utterance queue also grows fairly quickly. Try changing this parameter to 0.1 seconds instead and note the change in behavior.

Listing 8.3: A Simple iSay Example.

```
//----------------------------------------
ProcessConfig = iSay
{
  AppTick   = 4
  CommsTick = 4

  min_utter_interval = 1
  interval_policy    = from_end
  volume             = soft
}

//----------------------------------------
ProcessConfig = uTimerScript
{
  AppTick   = 4
  CommsTick = 4
```

```
  paused       = false
  reset_max    = nolimit
  reset_time   = all-posted
  delay_reset  = 0.25

  event        = var=SAY_MOOS, val="say={one}",   time=0.25
  event        = var=SAY_MOOS, val="say={two}",   time=0.5
  event        = var=SAY_MOOS, val="say={three},source=james", time=0.75
  event        = var=SAY_MOOS, val="say={four}",  time=1
  event        = var=SAY_MOOS, val="say={five},priority=100",  time=1.25
  event        = var=SAY_MOOS, val="say={six}",   time=1.5
  event        = var=SAY_MOOS, val="say={seven}", time=1.75
  event        = var=SAY_MOOS, val="say={eight}", time=2
  event        = var=SAY_MOOS, val="say={nine}",  time=2.25
}
```